UNIVAC 1050 DRUM SORT

PRELIMINARY USER DOCUMENT

CONTENTS

I. INTRODUCTION

A. General Description

B. Program Organization and Operation

C. Hardware Requirements

D. Software Requirements

II. SOURCE ORGANIZATION

A. Source Code Elements

B. Summary

C. Label Conventions

III. OWN CODING SECTIONS

A. First Pass Own Coding

B. Each Pass Own Coding

C. Last Pass Own Coding

IV. ERROR DISPLAYS

V. SAMPLE SORT PROGRAM

I. INTRODUCTION

A. GENERAL DESCRIPTION

The drum sort function for the UNIVAC 1050 System is provided by a set of source code procedures which, when assembled into a calling program, will: accept a file of unordered data, a record at a time; rearrange the records within the file according to specified relationships; and release the ordered file, a record at a time, for further processing.

1. Data Volume

The minimum amount of data that may be submitted is no data. The maximum amount that may be sorted in a single sort run is limited to the amount that can be contained in 64 full tracks of a Fastrand drum unit (64 x 64 x 64 sectors).

2. Record Characteristics

a. Record Size

All records to be processed by a given sort program must be of the same size. Any size from 1 through 1024 characters is acceptable.

b. Sort Key

The ordering relationship among records is based on a key, found in the same position in every record of a given sort. The key may consist of from 1 to 10 fields, each of which may be from 1 to 16 characters in length.

3. Record Selection

Any one of four types of comparison may be made on each key field to determine record selection within the sort.

Ascending binary: As the result of a binary comparison, the smaller field is chosen and the record to which it belongs is selected.

Descending binary: As the result of a binary comparison, the larger field is chosen and the record to which it belongs is selected.

Ascending decimal: As the result of a decimal comparison, the field which is smaller is chosen and the record to which it belongs is selected.

Descending decimal: As the result of a decimal comparison, the field which is larger is chosen and the record to which it belongs is selected.

A comparison is performed on the major key field of two records. If the fields are equal, the comparison for the next level key field is performed. This process continues until an inequality is found and a record can be selected. If all key fields of the two records are found to be equal, one of the records is selected arbitrarily.

B. PROGRAM ORGANIZATION AND OPERATION

Records are entered into the sort function by the user's first pass own coding (FPOC). FPOC obtains the records from some source, edits, alters, or otherwise processes them as desired, and delivers them by executing a sort subroutine. When the data has been ordered by the sort, the user's last pass own coding (LPOC) accepts the records by executing a sort subroutine and disposes of them as desired.

The sort program may also contain two other elements of own coding. Each pass own coding (EPOC) is executed during the collation passes, and allows the user to note, combine, delete, or otherwise process, records with duplicate keys. Permanent own coding comprises any instructions, data, or areas to be used by more than one other own coding section; for example, a PRNT routine and its associated output area. The permanent own coding resides in the low-order (permanent) portion of the sort program. (Refer to Figure 1).

A generated sort program functions with OPR, and, because of its segmentation, is loaded from either the drum or tape. After the first segment is loaded, the succeeding segments are entered automatically.

The program comprises 6 passes or phases:

parameter

dispersion, including FPOC

first collation
second collation
third collation
} Including EPOC, if any

merge, including LPOC

1. Parameter Pass

The parameter pass builds a table of information to be used by the subsequent passes. Based on the parameters of the various sort calls and the amount of memory actually available to the sort program, it calculates a block size for the current sort. The parameter pass also generates the sort tournament used by the dispersion pass.

2. Dispersion Pass

The dispersion pass moves records one at a time from an area in FPOC to the tournament area and from the tournament area to an output area. It then disperses sequences (strings) of ordered records to the drum area(s) assigned to the sort by FPOC. Strings are dispersed to a given drum track until one of the following conditions occurs: (1) the track is filled with data, (2) 64 strings have been written onto the track, or (3) FPOC signals the end of data. If condition (1) or (2) occurs, data is dispersed to the next track assigned to the sort; otherwise, the dispersion phase terminates, calling in the first collation phase. Up to 64 tracks may be used by the sort.

3. Collation Passes

The three collation passes merge the strings written by the dispersion pass to form 8 strings for input to the last (merge) pass. If the number of strings in a given track is greater than 8, the first collation pass is executed to reduce the number to 8. Then, if the number of strings in a given track is greater than 1, the second collation pass is executed to reduce the number to 1. When the first and second collation passes have been executed for all tracks involved in the sort, each track contains one string. If there are still more than 8 strings in the sort, the third collation pass reduces the number to 8. The merge pass is then called in. During the collation passes, each pass own coding, if any, is executed whenever two consecutive records with duplicate keys are detected. After performing any desired processing, EPOC returns control to the collation, exercising an option to either retain or delete the second record.

4. Merge Pass

The merge pass has as its input the 8 strings produced by the collation passes. As these strings are merged, the ordered "output" string is delivered, a record at a time, to LPOC. When all records have been delivered to LPOC, the merge pass terminates, passing control to LPOC's end-of-data section.

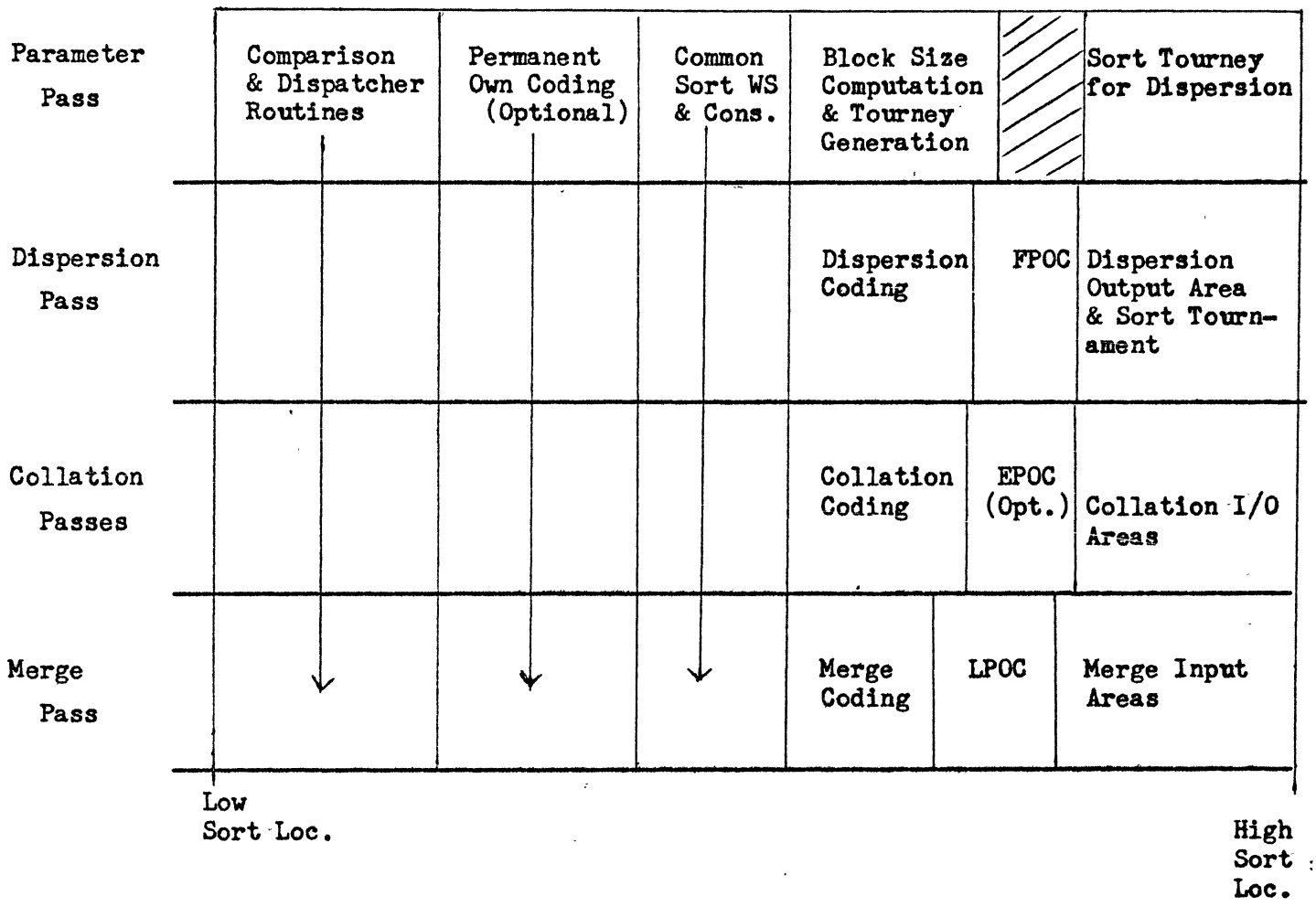

FIGURE 1. Organization of Sort Program Segments.

C. HARDWARE REQUIREMENTS

1. Fastrand Drum Unit

Either model I or model II of the Fastrand Drum Unit can be used. Depending on the volume of data to be sorted, from 0 to 64 tracks on a single unit will be required. The calling program specifies the tracks, or positions thereof, to be used.

2. Other Input/Output Devices

No other input/output devices are required by the drum sort, although the own coding sections may use any devices needed.

3. Store Requirements

A minimum of 12, 288 character positions is required.

D. SOFTWARE REQUIREMENTS

1. Operating System

The drum sort program will run with any OPR routine containing a Fastrand XF handler (that is, p6 of the OPR call was F or FL).

2. Other Drum Software

The drum sort program does not assume the absence or presence of any other drum software, and communicates directly with the drum XF handler of the operating system. The areas of the drum used by the sort are specified by FPOC, and it is up to the user's coding to ensure that these areas may be used legitimately. However, there is an optional sort call (DCHK) which will cause the sort to verify all drum addresses used against file usage tables or a FFILE packet.

The own-coding sections may call and use any drum software that they require.

II. SOURCE ORGANIZATION

A. SOURCE CODE ELEMENTS

1. BEGIN Line

The program ID in the LABEL field must be four characters of the form aaqr, where q and r must conform to the following rules:

(a) r must be a number 0 through 9, and

(b) if r is less than 7, q may be either a number or a letter; or

(c) if r is greater than 6, q must be a number 0 through 8.

In short, the program ID must be such that the segment ID's for the last three segments can be fabricated correctly.

The value in the OPERANDS field has the usual meaning. It is recommended that 5 be specified in order to maximize the amount of memory available to the sort. (Refer to DMEM call, (5) below.)

2. DSRT1 Call

The major function of the DSRT1 call is to provide the key comparison and drum XF dispatcher routines which will be used by the last 3 passes of the sort. The call specifies the number of key fields and the ordering relationship to hold for each. (The location of the key fields within each record is specified by the DKEYn calls, (4) below.)

LABEL	OP'N	OPERANDS
	DSRT1	p1, p2..., pn

The LABEL field is blank, and the OPERATION field is as shown.

p1 The number of key fields: 1 through 10.

p2 The ordering relationship for key field 1: B, for ascending binary; D, for ascending decimal; DB, for descending binary; or DD, for descending decimal.

p3 The ordering relationships for key fields 2 through 10, if any, specified in the same way as p2.

Parameter 2 specifies the major field, parameter 3 the first sub-major, and so on.

3. DCHK Call (Optional)

The optional DCHK call provides verification of all drum addresses accessed by the sort. Because the sort delivers XF's directly to the OPR drum handler, and because the addresses of the drum areas to be written into are provided by FPOC, it is strongly recommended that this call be present while FPOC is being debugged.

The file control routine (BDRUM, FDRMB, or FDRUM) used by the own coding must be in the permanent own coding section. If FDRUM is used, the pertinent FFILE packet must also be in the permanent own coding section. In addition, the file or files from which the addresses are taken must be open before FPOC delivers the first record to the sort.

LABEL	OP'N	OPERANDS
	DCHK	p1

The LABEL field is blank and the OPERATION field is as shown.

p1 If BDRUM or FDRMB is used, this parameter is BDRUM or FDRMB. If FDRUM is used, this parameter is the label of the FFILE call for the file involved. (In the former case, addresses will be checked against the file usage table by means of the WCHCK subroutine. In the latter case, addresses will be checked against the directory entry in the FFILE packet.)

4. DKEYn Calls

Each DKEYn call defines the length and position of a key field for which an ordering relationship was described in the DSRT1 call.

LABEL	OP'N	OPERANDS
	DKEYn	p1, p2

The LABEL field is blank. The OPERATION field is DKEYn, where n is 0 for key field 1, 1 for key field 2, and so on, up to 9 for key field 10.

p1 Length of the key field: 1 through 16.

p2 Address within record of the key field's LSC: 1 through 1024.

5. DMEM Call

The DMEM call specifies the limits, if any, on the number of store locations to be used by the sort program.

LABEL	OP'N	OPERANDS
	DMEM	[p1 [, p2]]

The LABEL field is blank and the OPERATION field is as shown.

p1 NOLIM specifies that the sort program is to use as many locations as are available to it when it is executed. (A blank OPERANDS field has the same meaning.) If p1 is other than NOLIM, it specifies the minimum number of locations to be reserved when the sort program is assembled.

p2 If p1 is other than NOLIM, p2, if present, specifies the the maximum number of locations to be used when the sort program is executed; otherwise, p2 is ignored.

Table 1 shows the effect that the BEGIN line and the parameters of the DMEM call have on the number of store locations used in the execution of the sort program.

TABLE 1. EFFECT OF BEGIN AND DMEM ON SORT STORE ALLOCATION

Legend:

A: Highest location in operating system.

B: Highest location in low relative program.

C: Lowest location in high relative program.

E: Highest location in store.

abs.: Absolute starting location from BEGIN line.

Sort Is	DMEM Call	BEGIN Line	Other Prog?	Number of Locations Used by Sort Program
Absolute	NOLIM or blank or p1	abs.	No	E - abs. + 1
	p1, p2	abs.	No	Smaller of: p2 E - Abs. + 1
Low Program (Relative)	NOLIM or blank or p1	3 or 5	No	E - A
			Yes	C - A
	p1, p2	3 or 5	No	Smaller of: p2 E-A
			Yes	Smaller of: p2 C-A
High Program (Relative)	NOLIM or blank	3	Yes	SORT CANNOT BE RUN
		5	Yes	E - B
	p1	3	Yes	p1
		5	Yes	E - B
	p1, p2	3	Yes	p1
		5	Yes	Smaller of: p2 E - B

6. DSIZE Call

The DSIZE call specifies the size of the records to be sorted.

LABEL	OP'N	OPERANDS
	DSIZE	p1

The LABEL field is blank and the OPERATION field is as shown.

p1 Record size: 1 through 1024.

7. Permanent Own Coding (Optional)

This section contains any elements of own coding which are to be referenced by more than one other own coding section. If a DCHK call is present, this section must contain the pertinent drum file control information.

8. DSRT2 Call

The DSRT2 call generates the parameter and dispersion passes.

LABEL	OP'N	OPERANDS
	DSRT2	

The LABEL and OPERANDS fields are blank, and the OPERATION field is as shown.

9. First Pass Own Coding

First pass own coding (FPOC) appears between the DSRT2 and DSRT3 calls. FPOC is entered when the dispersion pass is loaded by the parameter pass. It initializes the dispersion, allocates drum areas to the sort, delivers records to the sort, and terminates the dispersion by signalling end of data.

10. DSRT3 Call

The DSRT3 call generates the coding for the three collation passes and most of the coding for the merge pass.

LABEL	OP'N	OPERANDS
	DSRT3	pl

The LABEL field is blank and the OPERATION field is as shown.

pl — Blank specifies that no positive sequence check is to be performed during the collation.

Other than SEQCK or blank specifies that a positive sequence check is to be performed during each collation pass and that EPOC is to be executed whenever consecutive records with duplicate keys are detected. The value specified is the label of the EPOC subroutine.

SEQCK specifies that a positive sequence check is to be performed during each collation pass, but that there is no EPOC to be executed.

11. Each Pass Own Coding (Optional)

Each pass own coding (EPOC) is a closed subroutine which is loaded at the same time as the collation passes. It is executed whenever a collation pass selects a record having a key identical to that of the previously selected record. If n consecutive records having identical keys are selected, EPOC will be entered n-1 times.

12. DSRT4 Call

The DSRT4 call generates parts of the merge pass.

LABEL	OP'N	OPERANDS
	DSRT4	

The LABEL and OPERANDS fields are blank and the OPERATION field appears as shown.

13. Last Pass Own Coding

Last pass own coding (LPOC) appears between the DSRT4 and DSRT5 calls. LPOC is entered when the merge pass is loaded at the end of the collation passes. It initializes the merge pass and accepts records from the sort. Its end-of-data section is entered when the sorted data is exhausted.

14. DSRT5 Call

The DSRT5 call marks the end of the merge segment.

LABEL	OP'N	OPERANDS
	DSRT5	

The LABEL and OPERANDS fields are blank, and the OPERATION field is as shown.

15. END Line

The entry in the LABEL field of the END addresses the first LPOC instruction to be executed.

B. SUMMARY

The list on the next page summarizes the preceding description of the various elements of the sort program source code and dictates the order in which they are to appear. The numbers in the lefthand column refer to sections 1 through 15, above.

ORGANIZATION OF SORT SOURCE CODE

Number	Element	Comments
(1)	BEGIN line	Entry in OPERANDS field is absolute address, 3, or 5. Specification of 5 is recommended.
(2)	DSRT1 call	Defines number of key fields and ordering relationship for each. Generates key comparison and drum XF dispatcher routines.
(3)	DCHK call	Optional. Before accessing the drum, checks drum address against usage table or FFILE packet. Recommended for debugging.
(4)	DKEYn calls	One call for each key field, defining its length and position.
(5)	DMEM call	Defines limits, if any, to be placed on number of store locations used by sort. NOLIM or blank is recommended.
(6)	DSIZE call	Defines record size.
(7)	Permanent own coding	Contains any own code elements used by more than one of the other own coding sections. If DCHK call is present, must contain drum file control routine (and FFILE packet, if FDRUM used).
(8)	DSRT2 call	Generates parameter and dispersion passes.
(9)	First pass own coding	Initializes dispersion, delivers records to dispersion in store, terminates dispersion.
(10)	DSRT3 call	Generates collation and merge passes. Specifies whether positive sequence check is to be performed during collation. Specifies whether there is an each pass own coding (EPOC) section to be executed when records with duplicate keys are detected.
(11)	Each pass own coding	Optional. Executed whenever records with duplicate keys are detected. EPOC can direct sort to delete records.
(12)	DSRT4 call	Generates parts of merge pass.
(13)	Last pass own coding	Initializes merge, gets records from merge, terminates sort program when all records have been obtained.
(14)	DSRT5 call	Defines the end of the merge load (merge plus last pass own coding).
(15)	END Line	OPERANDS field specifies first LPOC instruction to be executed.

C. LABEL CONVENTIONS

Labels defined by the sort procedure start with the letter V. Since labels starting with U, V, W, X, Y, and Z are reserved for the software, the own coding labels should therefore start with one of the letters A through T.

III. OWN CODING SECTIONS

A. FIRST PASS OWN CODING

The first pass own coding section (FPOC) initializes the dispersion of the sort, delivers records to the dispersion, and terminates the dispersion when there are no more records to be delivered. FPOC also allocates one or more drum areas to be used as scratch areas by the sort.

1. Initialization

The first FPOC instruction to be executed must be labelled VBEGN. Control will be transferred to it when the dispersion segment is loaded.

Before delivering the first record to the sort, FPOC must initialize the dispersion and allocate the first drum area to it. This is done by executing a JR VSTRT, with the beginning drum address of the area in the 4 LSC of AR1, and the number of sectors in the 3 LSC of AR2. When control is returned, the dispersion has been initialized.

2. Delivering Records

Records are delivered to the sort in store and one at a time. To deliver a record, FPOC places the absolute address of the MSC of the record into index register 5, and executes a JR VPUT. When control is returned, the record has been transferred to an area in the sort.

3. Allocating Drum Areas

Once FPOC has initialized the dispersion, it may allocate other drum areas to the sort at any time. A drum area is allocated by placing the beginning drum address in the 4 LSC of AR1, the number of sectors in the 3 LSC of AR2, and then executing a JR VLIST.

If the area is accepted by the sort, control is returned to the second instruction following the JR. If the area is not accepted, control is returned to the first instruction following the JR. This error exit from the VLIST will not prevent further accesses to VLIST nor will it interfere with the normal operation of the sort.

When the error exit is made from VLIST, an error indication is set in location 0.

Location 0	Error
01	The unit of the drum area's beginning address differs from the unit of the area allocated in the initialization procedure.
02	The drum area is too small to be used by this sort. (The minimum size is found in VGGKØ.)
03	The drum area is situated on two units.
04	The list of allocated, but as yet unused, drum areas is currently full. Try again later.

The following considerations apply to the allocation of drum areas.

(1) Up to 64 areas may be allocated at one time.

(2) The areas are used by the sort on a last-listed first-used basis.

(3) The areas need not be in sequence with respect to their drum addresses. However, if two or more areas in a given track are to be listed, it is to the user's advantage to list them consecutively in time (but not necessarily in head-and-sector sequence).

(4) If an area encompasses two or more track addresses, the portion in each different track will be treated as a separate area (except in the sense of note (1)).

To illustrate, consider the following areas listed with VLIST:

(a) 010000 sectors, starting at 00105432;

(b) 01100 sectors, starting at 00116000; and

(c) 0203 sectors, starting at 00115500.

The sort would treat these portions as two areas, the first comprising three subareas:

(a) 0203 sectors, starting at 00115500;

(b) 01100 sectors, starting at 00116000;

(c) 05432 sectors, starting at 00110000; and

the second area comprising 02346 sectors, starting at 00105432.

4. Termination

When FPOC has no more records to deliver to the sort and has finished its processing, it executes a J VEND. The dispersion will terminate and the first collation pass will be loaded.

5. Index Register Assignment

(1) Before the JR VSTRT is executed, all index registers are free.

(2) When control is returned from VSTRT, index registers 1 and 4 have been set and must not be altered by FPOC.

(3) Once a JR VPUT has been executed, the status of the index register is as follows:

Index Register	Altered by VPUT?	May FPOC Alter?
1	Yes	No
2	Yes	Yes
3	Yes	Yes
4	Yes	No
5	No	Yes
6	No	Yes
7	No	Yes

(4) Executing a JR VLIST will not alter the index registers.

(5) The index registers will not be preserved from pass to pass.

6. Transfer Tetrads

The transfer tetrads will be altered every time that a JR VPUT is executed. They may be altered by FPOC.

B. EACH PASS OWN CODING

Each pass own coding (EPOC) is a closed subroutine which is entered when a collation pass detects consecutive records with duplicate keys.

When EPOC is entered, index register 2 contains the absolute address of the first character of the current sort record. Index register 1 contains the absolute address of the first character of a record which has a duplicate key. After performing any desired processing (excluding, of course, key alteration), EPOC returns control to the sort. If EPOC jumps to the recorded return address, the record that was addressed by index register 1 will be retained. If EPOC jumps to the recorded return address plus 5, the record that was addressed by index register 1 will be deleted from the sort.

EPOC may alter any or all index registers while it has control. Between entrances to EPOC, all index registers except number 4 will be altered by the collation coding.

It should be noted that the execution of EPOC does not guarantee that all duplicate keys will be detected, since it is possible, depending on the number and distribution of strings produced by the dispersion, that some of the data will not pass through the collation phases (that is, some data will go directly from dispersion to merge). In fact, a particular running of the sort program may bypass EPOC completely.

C. LAST PASS OWN CODING

Last pass own coding (LPOC) is entered when the merge segment is loaded. It initializes the merge, obtains the ordered records from the merge, and terminates itself when all records have been obtained.

1. Initialization

The label of the first LPOC instruction is to be executed when it appears in the OPERANDS field of the END line. Control will be transferred to it when the merge segment is loaded. Before obtaining the first record from the sort, LPOC must initialize the merge by executing a JR VINLP. When control is returned, the merge has been initialized.

2. Obtaining Records

Records are obtained from the sort in store and one at a time. To obtain a record, LPOC executes a JR VGET. When control is returned, the absolute address of the first character of the selected record is in index register 1.

3. End of Data

When a JR VGET is executed and there no records remaining, the sort transfers control to a LPOC instruction labelled VXEND which should be the beginning of LPOC's termination section.

4. Index Register Assignment

VINLP and VGET alter all the index registers except for numbers 3 and 4. LPOC may alter any index register.

IV. ERROR DISPLAYS

All error displays in the sort are of the form 0650ee, where ee is a unique error code. Displays not listed here originate in the own coding, the operating system, or DLL.

Pressing PROGRAM START after any display will cause the sort to jettison itself.

1. Parameter Pass

DISPLAY	EXPLANATION	ACTION
065001	Merge load will be too large for memory available.	
065002	Collation load will be too large for memory available.	Depending on cause, decrease own coding, remove DMEM restrictions and reassemble, or try running singly with smaller OPR.
065003	Dispersion load will be too large for memory available.	
065004	Not enough memory available for I/O in merge and/or collation passes.	
065005	Same as 065004.	
065006	Not enough memory available in dispersion for tournament and output areas.	
065010	Invalid program ID.	Change BEGIN line and reassemble, or alter segment ID's in object program.

2. Dispersion Pass

DISPLAY	EXPLANATION	ACTION
065020 065021 054022 065023 065024	Not enough drum area allocated to contain the data.	Run again with less data or more drum area.
065025	Data has exceeded capacity of sort (more than 64 tracks).	Run again with less data.
065031	Program error in VSTRT.	Run again.
065032	Drum area given to VSTRT too small.	Run again giving a valid area to VSTRT.
065033	Drum area given to VSTRT on 2 units.	
065034	Program error in VSTRT.	Run again.
065065	Unrecoverable drum error.	Run again.
065066	Out of bounds drum address. [1]	Run again with valid drum area.

3. Collation Passes

065065	Unrecoverable drum error.	
065066	Out of bounds drum address. [1]	
065070	Preselection error.	Run again.
065071	Block count error.	
065072	Sequence check error. [2]	

1. Error detected only when DCHK call has been used.
2. Error detected when sequence check specified in DSRT3 call.

4. Merge Pass

DISPLAY	EXPLANATION	ACTION
065065	Unrecoverable drum error.	
065066	Out of bounds drum address. [1]	Run again.
065070	Preselection error.	
065071	Block count error.	

1. Error detected only when DCHK call has been used.